Hierarchical and Dynamic Information Management Framework on Grid Computing

Eun-Ha Song, Yang-Seung Jeon, Sung-Kook Han, Young-Sik Jeong

Dept. of Computer Eng., Wonkwang Univ., 344-2 Shinyoung-Dong,
Iksan, 570-749, Korea
{ehsong, globaljeon, skhan, ysjeong}@wku.ac.kr

Abstract. This paper presents a GridIMF framework that provides support for adaptive grid services in response to the change of resource supplies and demands in a dynamic computing environment. The framework features a 3-tier hierarchical resource management structure. At the top is a global information manager that serves as a service broker between resource requesters and providers. Resource providers form virtual organizations, each of which is controlled by a local resource manager. The resource managers schedule the execution of tasks adaptively for efficiency and fault tolerance according to the dynamic resource availability information. The framework provides a common API through an application proxy. The proxy decouples grid applications from the implementation details of the framework.

1 Introduction

Grid is in the center of Virtual Organizations and the member of Virtual Organizations is an environment using not only a computing but also distributed computation or information resource as a virtual computer. In order to solve a grand challenge problem inside this paradigm, it is required to decide a policy or mechanism which can make the approach of grid[1][2]. Grid computing environment should take the feature that has unlimited number of resources, is spread into the area and organization and is heterogeneous system and the state of resources is variable. Grid User needs to the efficient grid resource management by integrating such resources in order to use them consistently[3][4][5]. The construction of framework which can support the upper level of requirement and include the facility to control grid work and information representation of grid resource is also needed in the grid computing environment.

Grid information includes metadata composing grid resource, grid resources and virtual organization which forms group of grid resources[6]. This paper adapts to variableness of grid information and establishes GridIMF(Grid Information Management Framework) which supports the requirement of user. GridIMF designs 3-tier

* This paper was supported by Wonkwang University in 2004.
* Corresponding Author : Young-Sik Jeong

hierarchical information management model classifying role and management policy of grid information functionally and logically. This provides scalability of grid information. GridIMF offers effective grid information management by designing the remote object reference for directly delivering an information saving model pursuant to the definition of an entry structure as well as computation results according to communication model pursuant to the definition of a connection object and a protocol object. GridIMF supports solubility of service with optimum selection of virtual organization and auto-recovery strategy in the virtual organization. It also improves the performance with adaptive performance-based task allocation method which supports load balancing and fault tolerance inside virtual organization. Grid users have different workload, requirement and job specification. This paper provides GridIMF and application proxy which divides the structure having minimum relation between them. Finally, it analyzes adaptability and executability of grid by applying grid application in GridIMF[7][8][9].

2 Related works

The current study related to grid is actively on the progress in research institutions over the world and Globus Toolkit is the most representing model[10]. Globus Toolkit connects and manages diversified heterogeneous resources of the lower layer in order to use them in the grid job and provides necessary service required in upper layer. Globus Toolkit is not a single system which cannot be separated but suggests the necessary service in grid as an independent element. It is very similar to GridIMF which was established in this paper from the point of view of reducing degradation between different systems and providing the integration function. But it is inefficient in the matter of management for metadata as it has the uniformed change and delivery cycle which ignore the characteristic of data. Thus, this Globus Toolkit can cause the degradation of grid performance.

Table 1. Globus vs. GridIMF

System / Activity Functions	Globus	GridIMF	Remark
Scalability of grid	●	●	3-tier hierarchical resource management model
Dynamic VO configuration	▲	●	
Availability for service	●	●	Optimum virtual organizations choosing method and Auto-recovery method
Deletion of replication operation at fault	×	●	
Resource state monitoring	●	●	PDH Library, Virtual Observer
Load balancing strategy	▲	●	Performance-based Task Allocation Method
Fault tolerance strategy	▲	●	
Independency among application	●	●	Application Proxy
User access interface	●	▲	Java Applet

(●: support, ▲: a partial support, ×: nonsupport)

GridIMF modeled in this paper accepts the information service of Globus Toolkit and service function of resource management as much as possible. GridIMF provides the Globus toolkit's MDS and GRAM supporting function with GVMS(Grid Virtual Management System) and GRMS(Grid Resource Management System). Service of GVMS considers scalability of grid by hierarchical node structure. User minimizes the load with optimum selection of virtual organization for the use of grid resource. Virtual organization suggests virtual organization auto-recovery strategy technology for server fault. GRMS service periodically receives the state of metadata included in local resource through Virtual Observer and accepts the load balancing and fault with adaptive performance-based task allocation method which operates scheduler in the virtual organization. Especially, GridIMF has the additional monitoring technology to understand the state of grid information in real-time and share and utilize them efficiently. Table 1 shows the comparison of Globus toolkit and supportive system which was presented in this system.

3 Grid Information Management Framework

3.1 Function Model of the GridIMF Components

GridIMF divides the components of grid into RR(Resource Requester), RP(Resource Provider), LRM(Local Resource Manager) and GIM(Global Information Manager) depending on the intention and purpose of participation. Fig. 1 shows the structure of architecture layer and control stream of suggested GridIMF components.

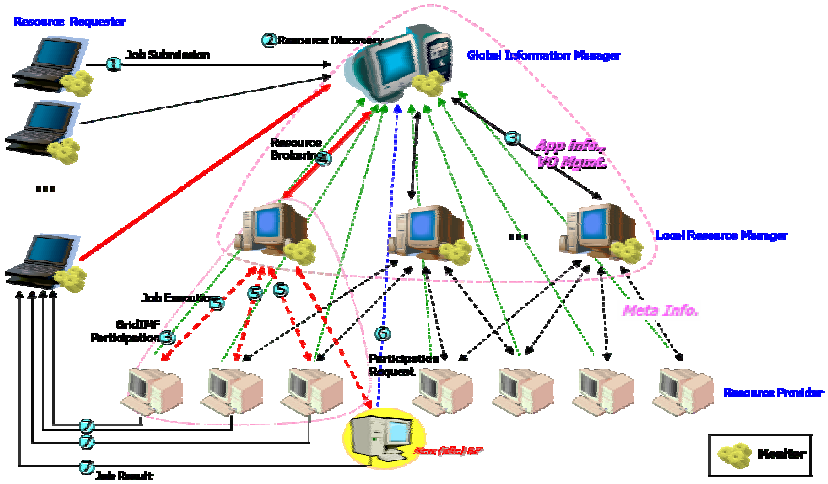

Fig. 1. Structure of layer and control stream of GridIMF components.

RR is the grid user and requests job submission to GIM. RR can request several jobs at the same time, only receives the corresponding result of requested job and

does not participate in computation. RR plays a role of interface delivering information of independent application. GIM is the top level manager and is a kind of resource brokering proxy connecting between resource and user who wants to use grid in remote. GIM manages the connection by providing common interface of components and executes job submission and control management as proxy. GIM is a super scheduler which brings the list of LRM satisfying based on job specification of specific application and permits the connection. LRM gets the delegation collecting resource and allocating task for requested job. LRM selects RP through resource discovery, intermediate the connection between RP and RR and makes core scheduling between remote resources. RP is a group which permits the execution for a part of task instead of huge scale of job and collects the information of metadata. RP delivers the executable resource, performs the operation and transmits the result.

3.2 GridIMF Communication Model

The components of GridIMF require to consistent in receiving data, command or query. For this, communication model is defined to provide activation of data exchange between information and active association of grid information. GridIMF communication model can be classified into Connectivity Object for connection between recourses and Protocol Object for generating and translating requirements such as command or query.

Depending on the hierarchical information management, LRM is a client for the position of GIM and performs two roles as a server for the position of RP. In order to prevent replication, GridIMF defines these connections as session with the complexity of lower communication. A session is classified into SessionProvider which processes the initial connection of components and has the server socket, ManagerSession which manages the communication of server side and Session which manages the communication of client side.

In communication layer, commands or requests delivered to the components go through the connectivity object in lower. Data transmitted through the session should be changed into a form recognizing the components for the transmission. Protocol object transmits by classifying into RR Protocol Data Parser, LRM Protocol Data Parser and RP Protocol Data Parser in each component. Delivered message defines common header and is used as monitoring element.

3.3 GridIMF Information Storage Model

Grid information can be widely divided into local and global resource. The characteristic of resources varies into its state, value, use, structure, etc. It generates entry for the function and use in order to prevent replication storage of grid information and make the consistency of use.

UpperEntry is defined as control of all entries and has generalized function having the concept. That is, UpperEntry gives entry_id and entry_name generated. Concertized entry is used as connection control, scheduling and monitoring information through the obtained identification information. EntryTable categorizes and stores

same conceptual components into the form of Hashtable. ApplicationEntry stores the information of application independently. The execution of application is related to RR, LRM and RP. RR records RRAppUI for the definition of application user interface as RP records for ProProxyInterfaceName processing common application proxy and LRM records for SchedulerName taking the reference data and make the scheduling. RREntry records ActiveApplication which is the name of application that currently corresponding RP executes and TaskResult which is the name of class for receiving the result of operation by remote object reference. LRMEntry records TotCpuSpeed, TotCpuNum, NumOfRP, etc. which are the benchmarking information of RPs. RPEntry is used as monitoring element and records ResourceFactor which is the element of metadata of RP, StateOfRP which is the state information and SizeOfTask which is the workload.

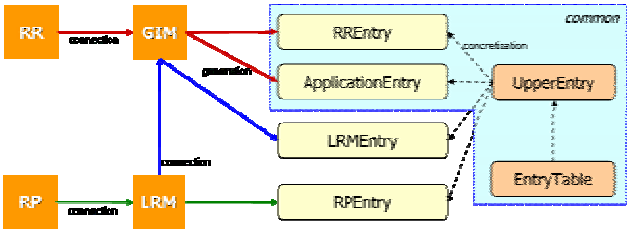

Fig. 2. Structure of GridIMF Entry.

3.4 GridIMF Remote Object Reference

Remote object reference is the mechanism reducing the load of LRM by transmitting the result of RP to RR directly. LRM generates RefBinder which is the list of RREntry and then binds the remote interface by exchanging it into RRRefBroker. The result of binding looks up RR and RP related to LRM. RR transmits remote object reference value to LRM through the interface of RemoteTaskResult with the class of TaskReceiver which is the class receiving the result. RR registers remote result processing class from RefBinder class generated by LRM and RR receives the class from RRReferBroker and sends the task result to RR. Fig. 3 is the interface of LRM and RR side defined for remote object reference.

```
public interface GridIMF_RRRefBroker extends Remote{          // the LRM side Interface
  public void GridIMF_SetRRRemoteReference(int iRRID, GridIMF_RemoteTaskResult
                                    nRRReceiverRef)  throws RemoteException;
  public void GridIMF_SetRRRemoteReference(GridIMF_RemoteTaskResult
                                    nRRReceiverRef)   throws RemoteException;
  public GridIMF_RemoteTaskResult GridIMF_GetRRRemoteReference(int iRRID)
                                    throws RemoteException;
  public GridIMF_RemoteTaskResult GridIMF_GetRRRemoteReference()
                                    throws RemoteException;
}
public interface GridIMF_RemoteTaskResult extends Remote {     // the RR side Interface
```

```
    public void SetProcessJobResult(
            int nRPID,          // RP ID
            int iTaskID,        // Task ID
            String sResult,     // Job Result
            int nStreamSize,    // Stream Data Size
            byte btStream[]   // Stream Data
        ) throws RemoteException;
}
```

Fig. 3. Interface of LRM and RR side for remote object reference

4 GridIMF Dynamic Information Management

4.1 Task Brokering Rule

Virtual organization has different policy, purpose and size. Grid application is finished its task by virtual organization with the scheduling. Therefore, the performance of application is the selection of virtual organization appropriate to the task. GridIMF defines 4 kinds of factors for selecting optimum virtual organization:

- Possibility of application execution: One application is assumed to be executed by one virtual organization. A single LRM is possible to execute various applications independently. GIM provides a list of application which is already executed and should be executed. LRM selects application which can be executed. The first stage selection depends on LRM which knows the capability and size of its virtual organization.
- Idle State: GIM senses the wait state by LRM virtual observer that is going to execute the requested task and then decides the possibility of execution.
- Application Power: The operation result of grid application is recorded as log file. The attribute of log file is the application name, execution time, LRM information, number of LRM and RP, total amount of static CPU speed, etc.
- LRM Performance Index: This performance index can be obtained by the analysis of application log file. Performance index selects a value close to the expected value in proportion to the average speed of CPU and number of nodes for executing requested application.

4.2 LRM State Control and Auto-Recovery Strategy

GIM is the manager of dynamic LRM so that transmission of control message is frequent related to LRM. GridIMF makes LRM virtual observer that gathers information of LRM and plays communication broker, and separates operation and control. LRM virtual observer is the monitor which makes scheduling the task and manages RPs inside virtual organization.

LRM auto-recovery method senses the LRM fault and secures the availability of service by obtaining the corresponding resource from another LRM. RR can be possible to use the grid information and to change the specification of executing application until the connection of GridIMF is disconnected. The fault information of LRM is accomplished by LRM virtual observer. When LRM fault is found, it looks for a new LRM in idle state. RR sends address, remote object reference and task table for the reference and operation is automatically executed by a new LRM. When the search for LRM is failed, RR is in wait state and added into the waiting list. The management of RR task table and remote object reference do not have replicated operation of LRM which generated fault.

4.3 Task Allocation Algorithm

GridIMF makes the counter measure and suggests DPTA(Dynamic Performance-based Task Allocation) and APTA(Adaptive Performance-based Task Allocation) for minimizing the cost of resource. Table 2 is the suggested task allocation method.

Table 2. Task Allocation Method of GridIMF

Task Allocation Method	Performance Evaluation & Reallocation Factor	Remark
Dynamic Performance-based Task Allocation	▪ CPU Speed ▪ Job History	▪ Fault Detection ▪ Append Processor ▪ State Monitoring
Adaptive Performance-based Task Allocation	▪ CPU Speed ▪ CPU Usage ▪ Proportional Factor	▪ Fault Detection ▪ Append Processor ▪ Realtime State Monitoring

DPTA is a method allocating and reallocating task according to the ratio of performance by considering RP performance. Performance index use CPU speed which are the static resource factor of RP and job history. Assuming total job amount for the application is $TotJobSize$ and CPU speed value of RPs is $\{sRP_0, sRP_2, \cdots, sRP_{NumRP-1}\}$, the job size of random RP_i is allocated as follows.

$$JobRP_i = \frac{sRP_i}{\sum_{k=0}^{NumRP-1} sRP_k} \times TotJobSize, \qquad i = 0 \sim NumRP - 1$$

The factor of job history is the factor measuring expectation of each RP performance and dynamically expects the job size per performance. The size of reallocated job for RP is as follows.

$$\mathrm{Re}AllocJobRP_i = pRateIncomRP_i \times (JobIncomRP_j - ComJobIncomRP_j)$$

$$pRateIncomRP_i = \frac{sComRP_i}{sComRP_i + sIncomRP_j \times \frac{ComJobIncomRP_j}{JobIncomRP_j}}$$

where, $sComRP_i$: Finished RP performance for the allocated job

$sIncomRP_j$: Unfinished RP performance for the allocated job

$JobsIncomRP_j$: Unfinished size of initially allocated RP job for allocated job

$ComJobsIncomRP_j$: Size of job performed by RP which did not finish the allocated job

$pRateIncomRP_j$: RP performance index occurring performance change

APTA is a method which monitors the static and dynamic performance of RP and allocates the job adaptively. This establishes a standard based on the performance executing real allocated task not a physical performance for the RP performance. RP performance value applies proportional factor and CPU speed. Proportional Factor(PF) is a standard value as CPU usage rate used by RP user or internal system differs in the degree of total execution time depending on the range. Table 3 is the proportional factor in different section.

Table 3. Proportional Factor in different section

Section	User CPU Usage	PF(Proportional Factor)
A	0~10%	2.10
B	10~20%	1.91
C	20~30%	1.67
D	30~40%	1.40
E	40~60%	1.09
F	60% over	1.00

Assuming CPU speed value of RPs is $\{sRP_0, sRP_2, \cdots, sRP_{NumRP-1}\}$, the expectation of total CPU usage is $CPUUsageAvg$ at the time of reallocation and expectation of job processor usage rate is $JobUsageAvg$, the performance value of random RP_i is as follows.

$$RP_i Performance = (100 - (CPUUsageAvg - JobUsageAvg)) \times PF \times sRP_i \times 0.01$$

APTA uses RPPerformance value not the job history information and its reallocation is similar to DPTA. Reallocated job size of RP is as follows.

$$\mathrm{Re}\, AllocJobRP_i = pRateIncomRP_i \times (JobIncomRP_j - ComJobIncomRP_j)$$

$$pRateIncomRP_i = \frac{pComRP_i}{pComRP_i + pIncomRP_j \times \dfrac{ComJobIncomRP_j}{JobIncomRP_j}}$$

where, $pComRP_i$: value of RP which finished the allocated job

$pIncomRP_j$: value of RP which did not finish the allocated job

Fig. 4 is the comparison of task allocation method that user gives random CPU usage rate to 4 RPs with different specification. For the case if user CPU usage rate is below 10%, there is no difference in task allocation methods. This implies that performance changes almost did not in RPs and it did not influence on the task performance rate. On the contrary, the case of user CPU usage rate increasing to 20~30% and 30~40%, the total job performance time of APTA considering total CUP usage rate is

low. That is, user CPU usage rate of RP influenced on job performance rate of total CPU usage rate.

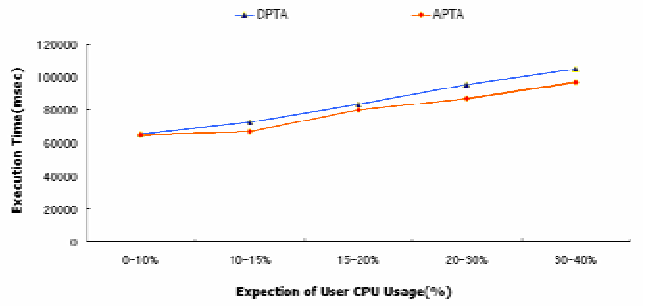

Fig. 4. Comparison of task allocation method according to the change of user CPU usage rate

5 GridIMF implementation and performance result

5.1 Application Architecture

Application is implemented with the verification of strategy suggested in GridIMF. Grid user has different request depending on application so that the application architecture is structurally divided only with minimum correlation.

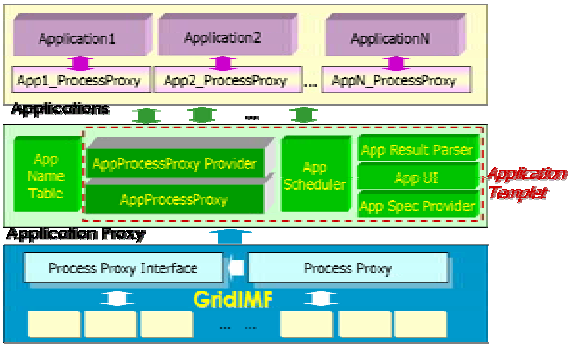

Fig. 5. Application Architecture of GridIMF

For the mechanism minimizing interdependence between different application and GridIMF, Application Proxy is designed as shown in Fig. 5. Application Proxy is a standardized common API and includes core function for interacting between application and GridIMF. Application proxy is an Application Name of plug-in for accessing specific application and the object is generated dynamically by Application Templet.

5.2 Implementation of Application and Result of Performance

Implemented application is Mandelbrot's Fractal Image Processing depending on LSM coloring method as shown in Fig. 6. Task allocation algorithm is APTA and new two RPs participate in operation one after the other while the job is performed by initial two RPs. (c) shows the occurrence of reallocation by additional RP and the coloring method differed for comparison. (d) is an example of processing reoperation changed by request of user. When a specific area is set up by drag function, the specification of application is changed and the corresponding job is performed again.

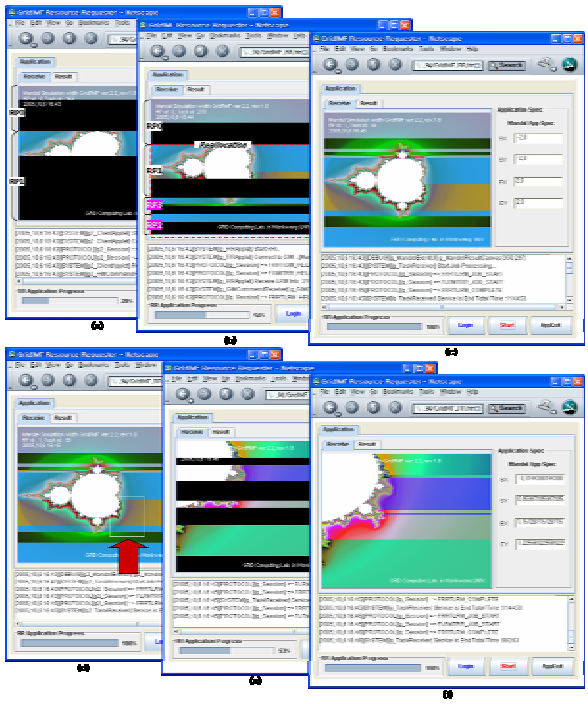

Fig. 6. Process of fractal image generation

6 Conclusions

Grid computing environment integrates resources, composes grid information and becomes a single huge system by forming one virtual organization for the purpose and characteristic. This paper established GridIMF in order to use the grid information consistent used by user and implemented a real grid application.

GridIMF designed 3-tier hierarchical information management mode dividing into GIM, LRM, RP and RR depending on the characteristic of grid information. For the efficient operation of components, management domain was classified into GVMS

and GRMS. GVMS suggested optimum virtual organization selection method for job brokering of application and virtual organization. Also, it suggested LRM auto-recovery strategy considering request change of user and replication of operation even though fault of virtual organization occurred. GRMS suggested APTA task allocation method for the change, participation and fault of grid information state, and evaluated its performance. Also application proxy was designed for satisfying the characteristic of grid application and supporting minimum code modification as standardized API. Then, Fractal image generation was applied for analyzing performance of GridIMF and grid application. The main contributions of this paper are as followed; scalability(3-tier hierarchical resource management structure), adaptive(dynamic virtual organization, task allocation), availability of service, deletion of replication operation and independent of applications.

With the further study, interdependent application between detailed jobs or other specific application can be implemented by adding the user access interface function which can process detailed job procedure for user in the point of GridIMF access suggested in this paper.

References

1. I. Foster, C. Kesselman, "The Grid: Blueprint for a New Computing Infrastructure", Morgan Kaufmannn, 1999.
2. I. Foster, C. Kesselman, S. Tueche, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations", International Journal Supercomputer Applications, Vol. 15, No. 3, 2001.
3. I. Foster, "The Grid: A New Infrastructure for 21st Century Science," Physics Today.org, 2002.
4. Dr. Bernhard R. Katzy, "Design and Implementation of Virtual Organizations," University St. Gallen and Erasmus University Rotterdam
5. Rashmi Bajaj and Dharma P. Agrawal, Fellow, "Improving Scheduling of Tasks in a Heterogeneous Environment," IEEE Trans. Parallel and Distributed Systems, Vol. 15, No. 2, pp. 107-118, 2004.
6. A. Takefusa, H. Casanova, and S. Matsuoka, F. Berman, "A Study of Deadline Scheduling for Client-Server Systems on the Computational Grid," High Performance Distributed Computing, 2001. Proceedings 10th IEEE International Symposium, Aug. 2001
7. Rajkumar Buyya, Steve Chapin, and David DiNucci, "Architectural Models for Resource Management in the Grid," Grid Computing GIRD 2000. First IEEE/ACM International Workshop Bangalore, India, pp. 20-33, 2000.
8. Haban, D. Shin, K. G, "Application of RealTime Monitoring to Scheduling Tasks with Random Execution Times," IEEE Transactions on Software Engineering, Vol. 16, pp. 1374-1389, Dec. 1990
9. D. Angulo, I. Foster, C. Liu, and L. Yang., "Design and Evaluation of a Resource Selection Framework for Grid Applications," Proceedings of IEEE International Symposium on High Performance Distributed Computing (HPDC-11), Edinburgh, Scotland, July 2002.
10. I. Foster, C. Kesselman, "Globus: A Metacomputing Infrastructure Toolkit, " International Journal supercomputer Applications, Vol. 11, No. 2, pp. 115-128, 1997